

# On the Convergence Speed of Turbo Demodulation with Turbo Decoding

Salim Haddad, Amer Baghdadi, and Michel Jezequel

## Abstract

Iterative processing is widely adopted nowadays in modern wireless receivers for advanced channel codes like turbo and LDPC codes. Extension of this principle with an additional iterative feedback loop to the demapping function has proven to provide substantial error performance gain. However, the adoption of iterative demodulation with turbo decoding is constrained by the additional implied implementation complexity, heavily impacting latency and power consumption. In this paper, we analyze the convergence speed of these combined two iterative processes in order to determine the exact required number of iterations at each level. Extrinsic information transfer (EXIT) charts are used for a thorough analysis at different modulation orders and code rates. An original iteration scheduling is proposed reducing two demapping iterations with reasonable performance loss of less than 0.15 dB. Analyzing and normalizing the computational and memory access complexity, which directly impact latency and power consumption, demonstrates the considerable gains of the proposed scheduling and the promising contributions of the proposed analysis.

## I. INTRODUCTION

Advanced wireless communication standards impose the use of modern techniques to improve spectral efficiency and reliability. Among these techniques Bit-Interleaved Coded Modulation (BICM) with different modulation orders and Turbo Codes with various code rates are frequently adopted.

The BICM principle [1] currently represents the state-of-the-art in coded modulations over fading channels. The Bit-Interleaved Coded Modulation with Iterative Demapping (BICM-ID) scheme proposed in [2] is based on BICM with additional soft feedback from the Soft-Input Soft-Output (SISO) convolutional decoder to the constellation demapper. In this context, several techniques and configurations have been explored. In [3], the authors investigated different mapping techniques suited for BICM-ID and QAM16 constellations. They proposed several mapping schemes providing significant coding gains. In [4], the convolutional code classically used in BICM-ID schemes was replaced by a turbo code. Only a small gain of 0.1 dB was observed. This result may make BICM-ID with turbo-like coding solutions (TBICM-ID) unsatisfactory with respect to the added decoding complexity. On the other hand, authors in [5] have presented a technique intended to improve the performance of TBICM-ID over non Gaussian channels. The proposed technique, namely Signal Space Diversity (SSD), consists of a rotation of the constellation followed by a signal space component interleaving. It has shown additional error correction at the receiver side in an iterative processing scenario.

Constellation rotation enables to exploit higher code rates and to solve potential problems in selective channels while keeping good performance. It has been proposed for all constellation orders of Quadrature Amplitude Modulation (QAM). Combining constellation rotation with signal space component interleaving leads to significant improvement in performance over fading and erasure channels. It increases the diversity order of a communication system without using extra bandwidth. BICM coupled with SSD has been extensively studied for single carrier systems, e.g. in [6], [7] and the references therein.

Application of iterative demapping in this context has shown excellent error rate performance results particularly in severe channel conditions (erasure, multi-path, real fading models) [8]. In that work, LDPC channel coding was considered.

Nevertheless, most of the existing works have not considered these techniques from an implementation perspective. In fact, the application of the iterative demapping in future receivers using advanced iterative channel decoding will lead to further latency problems, more power consumption and more complexity caused by feedback inner and outer the decoder. Besides extrinsic information exchange inside the iterative channel decoder, additional extrinsic information is fed back as *a priori* information used by the demapper to improve the symbol to bit conversion. The number of iterations to be run at each level should be determined accurately as it impacts significantly, besides error rate performance, latency, power consumption, and complexity.

This work discusses the implementation efficiency of iterative receiver based on turbo demodulation and turbo decoding in order to achieve a gain in band-limited wireless communication systems. Convergence speed is analyzed for various system configurations to determine the exact required number of iterations at each level. Significant complexity reductions can be achieved by means of the proposed original iteration scheduling. Finally, an accurate normalized complexity analysis is presented in terms of arithmetic and memory access operations.

## II. SYSTEM MODEL AND ALGORITHMS

### A. System Model

The considered system uses one transmit and one receive antenna while assuming perfect synchronization. Fig. 1 shows a basic transmitter and receiver model using turbo demodulation and decoding. We denote by TBICM-ID-SSD a turbo BICM with iterative demapping coupled with signal space diversity.

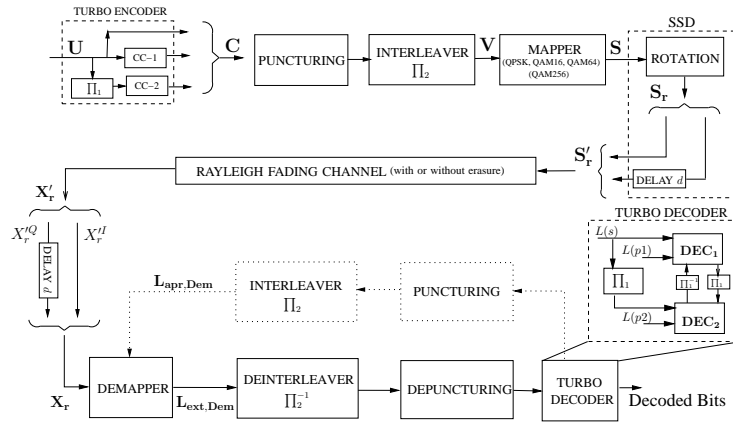


Fig. 1: System model with TBICM-ID-SSD.

On the transmitter side, information bits  $U$  which are called systematic bits are regrouped into symbols  $u_p$  consisting of  $l$  bits, and encoded with an  $l$ -binary turbo encoder. It consists of a parallel concatenation of two identical convolutional codes (PCCC). The output codeword  $C$  is then punctured to reach a desired coding rate  $R_c$ . The 8-state double binary ( $l = 2$ ) recursive systematic convolutional code (RSC code), adopted in the WiMax standard, is considered for the turbo encoder.

In order to gain resilience against error bursts, the resulting sequence is interleaved using an  $S$ -random interleaver  $\Pi_2$  with  $S = \sqrt{N/4}$ . Punctured and interleaved bits denoted by  $V$  are then gray mapped to channel symbols  $s_q$  chosen from a  $2^M$ -ary constellation  $X$ ,  $M$  is the number of bits per modulated symbol.

Applying the SSD consists first of the rotation of the mapped symbols  $s_q$ . The resulting rotated symbols are denoted by  $s_{r,q}$ . The performance gain obtained when using a rotated constellation  $X_r$  depends on the choice of the rotation angle. In this regard, a thorough analysis has been done for the 2nd-generation terrestrial transmission system developed by the DVB Project (DVB-T2) which adopted the rotated constellation technique. A single rotation angle [8] has been chosen for each constellation size independently of the channel type. Using these angles, and with LDPC code, gains of 0.5 dB and 6 dB were shown for Rayleigh fading channel without and with erasure respectively for high code rate [8]. The second step consists of signal space component interleaving. When a constellation signal is submitted to a fading event, its in-phase component  $I$  and quadrature component  $Q$  fade identically and suffers from an irreversible loss. A means of avoiding this loss involves making  $I$  and  $Q$  fade independently while each carrying all the information regarding the transmitted symbol. By inserting an interleaver (a simple delay for uncorrelated fast-fading channel) between the  $I$  and  $Q$  channels, the diversity order is doubled.

$s'_{r,q}$  symbols are then transmitted over a noisy and Rayleigh fast-fading channel with or without erasures. In fact, erasure events happen in single frequency network (SFN), as in DVB-T2 standard, due to destructive interferences. Each received symbol  $x'_{r,q}$  is affected by a different fading coefficient, an erasure coefficient, and additive Gaussian noise.

The channel model considered is a frequency non-selective memoryless channel with erasure probability. The received discrete time baseband complex signal can be written as:

$$\begin{aligned} x'_{r,q} &= h_q \cdot \rho_q \cdot s'_{r,q} + n_q \\ &= h'_q \cdot s'_{r,q} + n_q \end{aligned} \quad (1)$$

where  $h_q$  is the Rayleigh fast-fading coefficient,  $\rho_q$  is the erasure coefficient probability taking value 0 with a probability  $P_\rho$  and value 1 with a probability of  $1 - P_\rho$ .  $n_q$  is a complex white Gaussian noise with spectral density  $N_0/2$  in each component axes, and  $h'_q$  is the channel attenuation. Note that, at the receiver side, the transmitted energy has to be normalized by a  $\sqrt{1 - P_\rho}$  factor in order to cope with the loss of transmitted power.

### B. Max-Log-MAP Demapping Algorithm

At the receiver side, the complex received symbols  $x'_{r,q}$  have their Q-components re-shifted resulting in  $x_{r,q}$ . An extrinsic log-likelihood ratio  $L_{ext, Dem}(c_{p,q}/x_{r,q})$  is calculated for each bit  $c_{p,q}$  corresponding to the  $p^{th}$  bit of the received rotated and modulated symbol  $x_{r,q}$ . After de-interleaving, de-puncturing and turbo decoding, extrinsic information from the turbo decoder  $L_{ext, Dec}(c_{p,q})$  is passed through the interleaver, punctured and fed back as *a priori* information  $L_{apr, Dem}(c_{p,q})$  to the demapper in a turbo demapping scheme.  $L_{ext, Dem}(c_{p,q}/x_{r,q})$  is the difference between the soft output *a posteriori*  $L_{Dem}(c_{p,q}/x_{r,q})$  and  $L_{apr, Dem}(c_{p,q})$  at the demapper side, it is given by the expression below:

$$\begin{aligned} L_{ext, Dem}(c_{p,q}/x_{r,q}) &= L_{Dem}(c_{p,q}/x_{r,q}) - L_{apr, Dem}(c_{p,q}) \\ &= \log \left( \frac{Z_1}{Z_2} \right) \end{aligned} \quad (2)$$

$Z_{l(l=0,1)}$  can be expressed as:

$$Z_{l(l=0,1)} = \sum_{s_{r,j} \in X_{r,l}^k} e^{-A_q} \cdot \prod_{i=0, i \neq k}^{M-1} P(c_{i,q}) \quad (3)$$

where  $X_{r,l}^k$ , with  $l \in \{0,1\}$ , are the symbol sets of the constellation for which symbols have their  $i^{th}$  bit equal to  $l$ .  $P(c_{i,q})$  is the probability of the  $i^{th}$  bit of constellation symbol  $s_{r,q}$  computed through *a priori* information  $L_{apr, Dem}(c_{i,q})$ . Reducing the complexity of the expressions above can be performed by applying the max-log approximation. Thus, equation (2) can be written as:

$$L_{ext, Dem}(c_{p,q}/x_{r,q}) = \min_{s_{r,j} \in X_{r,0}^k} (A_q - B_{p,q}) - \min_{s_{r,j} \in X_{r,1}^k} (A_q - B_{p,q}) \quad (4)$$

where  $A_q$  and  $B_{p,q}$  are computed as follows.

$$A_q = \frac{h_q'^2}{\sigma^2} |x_{r,q}^I - s_{r,j}^I|^2 + \frac{h_q'^2}{\sigma^2} |x_{r,q}^Q - s_{r,j}^Q|^2 \quad (5)$$

$$B_{p,q} = \left( \sum_{i=0, c_{i,q}=1}^{M-1} L_{apr, Dem}(c_{i,q}) \right) - L_{apr, Dem}(c_{p,q}) \quad (6)$$

In fact, the above demapping equations are valid for both channel models (with or without erasures) through the use of  $h_q'$  coefficient (equation (1)).

These simplified expressions exhibit three main computation steps: (a) Euclidean distance computation referred by  $A_q$ , (b) *a priori* adder referred by  $B_{p,q}$ , and (c) minimum finder referred by the *min* operation of equation (4).

### C. Max-Log-MAP Decoding Algorithm

Following the demapping function at the receiver side, the turbo decoding algorithm is applied. The BCJR algorithm is considered for the Soft Input Soft Output (SISO) convolutional decoders. Using input symbols and *a priori* extrinsic information, each SISO decoder computes *a posteriori* probabilities. The BCJR SISO decoder computes first the branch metrics  $\gamma$ . Then it computes the forward  $\alpha_k$  and backward  $\beta_k$  metrics between two trellis states  $s$  and  $s'$  [9].

$$\alpha_k(s) = \max_{(s', s)} (\alpha_{k-1}(s') + \gamma_k(s', s)) \quad (7)$$

$$\beta_k(s) = \max_{(s', s)} (\beta_{k+1}(s') + \gamma_{k+1}(s', s)) \quad (8)$$

where

$$\gamma_k(s', s) = \gamma_k^{Sys}(s', s) + \gamma_k^{Parity}(s', s) + \gamma_k^{Ext}(s', s) \quad (9)$$

The soft output information  $so(d_k = c_p c_{p+1})$  and symbol-level extrinsic information  $z(d_k = c_p c_{p+1})$  of symbol  $k$  are then computed using equations (10) and (11). The extrinsic information, which is exchanged iteratively between the two SISO decoders, is obtained by subtracting the intrinsic information from  $so(d_k = c_p c_{p+1})$ .

$$so(d_k) = \max_{(s', s)/d(s', s)=d_k} (\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)) \quad (10)$$

$$z(d_k) = \max_{(s', s)/d(s', s)=d_k} (\alpha_{k-1}(s') + \gamma_k^{Ext}(s', s) + \beta_k(s)) \quad (11)$$

$z(d_k)$  can be multiplied by a constant scaling factor  $SF$  (typically equals to 0.75) for a modified Max-Log-MAP algorithm improving the resultant error rate performance.

Finally, in case of turbo demapping and only by one SISO decoder, the bit-level extrinsic information of systematic symbols  $c_p c_{p+1}$  are computed using equations (12) and (13). Similar computations are done for parity symbols  $c_{p+2} c_{p+3}$ .

$$L_{apr, Dem}(c_p) = \max[z(d_k = 11), z(d_k = 10)] - \max[z(d_k = 01), z(d_k = 00)] \quad (12)$$

$$L_{apr, Dem}(c_{p+1}) = \max[z(d_k = 11), z(d_k = 01)] - \max[z(d_k = 10), z(d_k = 00)] \quad (13)$$

These expressions exhibit three main computation steps: (a) branch metrics computation referred by  $\gamma_k$ , (b) state metrics computation referred by  $(\alpha_k$  and  $\beta_k)$ , and (c) extrinsic information computation referred by  $L_{apr, Dem}$  and  $z$ .

### III. TBICM-SSD AND TBICM-ID-SSD CONVERGENCE SPEED ANALYSIS

This section illustrates the impact of constellation rotation and iterative demapping on the convergence speed. Convergence speed designates the rapidity of the convergence of the iterative process.

EXIT charts [10] are used as a useful tool for a clear and thorough analysis of the convergence speed. They were first proposed for parallel concatenated codes, and then extended to other iterative processes. For iterative demapping with turbo decoding (TBICM-ID-SSD), authors in [5] have used this tool to analyze the iterative exchange of information between the different SISO components. In this system receiver with two iterative processes, the response of the two SISO decoders is plotted while taking into consideration the SISO demapper with updated inputs and outputs.

In this scheme,  $IA_1$ ,  $IA_2$ ,  $IE_1$ ,  $IE_2$  are used to designate the *a priori* and extrinsic information respectively for DEC<sub>1</sub> and DEC<sub>2</sub> (Fig. 1). Iterations start without *a priori* information ( $IA_1 = 0$  and  $IA_2 = 0$ ). Then, extrinsic information  $IE_1$  of DEC<sub>1</sub> is fed to DEC<sub>2</sub> as *a priori* information  $IA_2$  and vice versa, i.e.  $IE_1 = IA_2$  and  $IE_2 = IA_1$ . Since this EXIT chart analysis is asymptotic, infinite long BICM interleaver size should be assumed. The SISO decoder is represented by its transfer function:

$$IE = T(IA, E_b/N_0) \quad (14)$$

Extensive analysis for different  $E_b/N_0$  and different system parameters (modulation orders and code rates) has been conducted and gave similar results. Fig. 2 illustrates one of these simulations for QAM64, code rate  $\frac{4}{5}$ ,  $E_b/N_0=22$  dB, and erasure probability equals to 0.15. The transfer function of the turbo decoder is represented by the two-dimensional chart as follows. One SISO decoder component is plotted with its input on the horizontal axis and its output on the vertical axis. The other SISO component is plotted with its input on the vertical axis and its output on the horizontal axis. The iterative decoding corresponds to the trajectory found by stepping between the two curves. For a successful decoding, there must be a clear path between the two curves so that iterative decoding can proceed from 0 to 1 mutual extrinsic information.

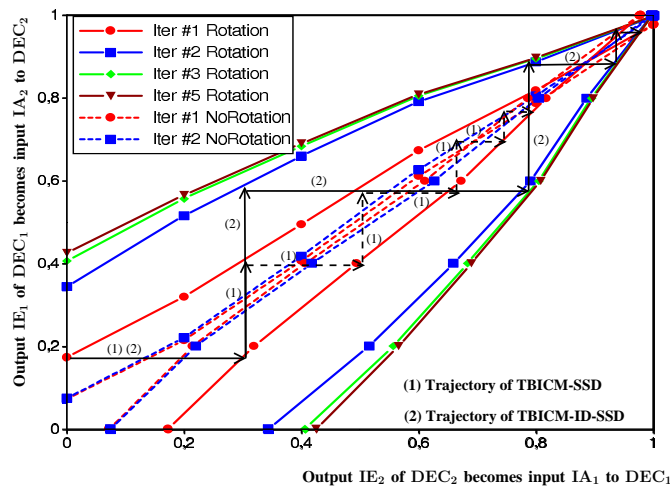


Fig. 2: EXIT chart analysis at an  $E_b/N_0=22$  dB of the dual binary turbo decoder for iterations to the QAM64 demapper. Code rate  $\frac{4}{5}$  is considered for transmission over Rayleigh fast-fading channel with erasure probability equals to 0.15.

The plain curves of Fig. 2 correspond to the EXIT charts for the case with rotated constellation. Meanwhile, the dashed curves correspond to the case with no rotation. Furthermore, the red curves correspond to non iterative demapping, e.g. SISO demapper executed once in feed forward scheme. Applying demapping iterations corresponds to the other colored curves in the EXIT charts of Fig. 2.

In this figure, we observe that the EXIT tunnel is wider for the rotated case than the one without. Furthermore, the tunnel is limited to that of one demapping iteration for the latter case. Thus, making more demapping iterations will not affect the convergence speed of non rotated constellation configurations. However the tunnel is enlarging (improving) until three demapping iterations using the rotated constellation. For TBICM-SSD, EXIT charts show a need of more than 6 turbo decoding iterations to attain convergence following the trajectory (1). Whereas 4 demapping iterations are sufficient following the trajectory (2).

Thus, in case of TBICM-ID-SSD, the iteration scheduling which optimize the convergence is the one that enlarge the EXIT tunnel as soon as possible. Analyzing the different tunnel curves in the EXIT figure shows that the tunnel is enlarging for each demapping iteration. Thus, the optimized scheduling is to execute only one turbo decoding iteration for each demapping iteration and then step forward to the next demapping iteration (enlarge the EXIT tunnel). This scheduling is the one adopted implicitly in [5]. Note that after the third demapping iteration, only a slight improvement in convergence is observed. Similar results have been found for all considered modulation orders, code rates and erasure coefficients. This result will be used in the next section to reduce the number of demapping iterations.

#### IV. REDUCING THE NUMBER OF DEMAPPING ITERATIONS

As mentioned in the previous section, the optimized profile applies one turbo code iteration for each demapping iteration. Thus, reducing the number of turbo demapping iterations will reduce the total number of iterations for the turbo decoder.

However, various constructed EXIT charts with different parameters show that after a specific number of demapping iterations, only a slight improvement is predicted. As an example, in Fig. 2 decoder transfer functions coincide with each other after 3 demapping iterations. However, one can notice that turbo decoding iterations must continue until that the two constituent decoders agree with each other. Thus, the number of demapping iterations can be reduced without affecting error rates, while keeping the same total number of turbo decoding iterations. This constitutes the basis for our proposed original iteration scheduling.

In fact, to keep the same number of iterations for the decoder unaltered, one turbo code iteration is added after the last iteration to the demapper for each eliminated demapping iteration. Fig. 3 simulates six turbo demapping iterations performing one turbo decoding iteration for each. Hence, six turbo code iterations are performed in total. This scheme is denoted as  $6IDem$ .

With the proposed iteration scheduling,  $5IDem\_1EIDec$  designates five demapping iterations (one turbo code iteration is applied for each) followed by one extra turbo code iteration.

Referring to Fig. 3, error rates associated to  $6IDem$  and  $5IDem\_1EIDec$  show almost same performances, while one feedback to the demapper is eliminated in the latter scheme. Similarly, for  $4IDem\_2EIDec$ , two feedbacks to the demapper are eliminated. A slight loss of 0.025 dB is induced. Eliminating more demapping iterations will cause significant performance degradation.  $3IDem\_3EIDec$  is closer to  $5IDem$  than to  $6IDem$ .

In fact,  $4IDem\_2EIDec$  represents the most optimized curve for the  $6IDem$  performance scheme as shown in Fig. 3. EXIT charts do not agree with this consideration at the first sight, three demapping iterations were sufficient to do the same correction as the eight iterations. EXIT charts are based on average calculations as many frames are simulated.

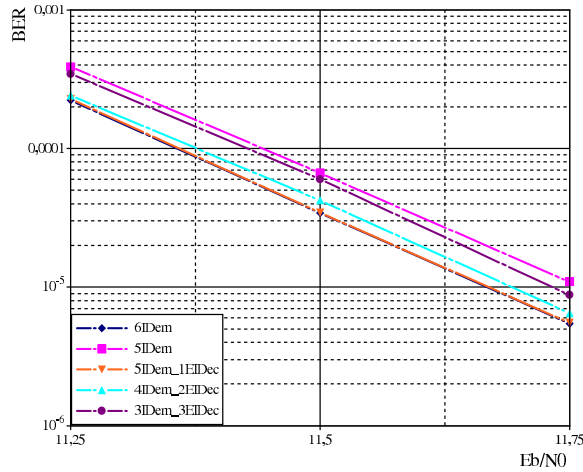


Fig. 3: BER performance comparison for TBICM-ID-SSD for the transmission of 1536 information bits frame over Rayleigh fast-fading channel without erasure. QAM64 modulation scheme with code rate  $\frac{2}{3}$  are considered.

Modulation scheme	Performance loss (dB)	
	Without Erasure	With Erasure
	$R_c = 6/7 \rightarrow R_c = 1/2$	$R_c = 6/7 \rightarrow R_c = 1/2$
QPSK	0.02 $\rightarrow$ 0.03	0.02 $\rightarrow$ 0.05
QAM16	0.04 $\rightarrow$ 0.06	0.04 $\rightarrow$ 0.08
QAM64	0.05 $\rightarrow$ 0.08	0.07 $\rightarrow$ 0.12
QAM256	0.07 $\rightarrow$ 0.10	0.09 $\rightarrow$ 0.15

TABLE I: Performance loss for different modulation schemes and code rates after 2 omitted demapping iterations.

The three demapping iterations represents the average number of demapping iterations needed to be sure that the two constituent decoders agree with each other. Making more demapping iterations will provide more error correction. Further simulations show performance loss of 0.02 dB to 0.1 dB and 0.02 dB to 0.15 dB for no erasure and erasure events respectively when the proposed scheduling is applied. Table I summarizes the reduced performance loss for different code rates and constellation orders after omitting two demapping iterations. These values were investigated for the worst case corresponding to *3IDem\_2EIDec* in comparison to *5IDem*. Note that for error floor region, simulations show almost identical BER performance if applying more than 3 demapping iterations. Furthermore, it is worth noting that with a limited-diversity channel model, omitting 2 demapping iterations leads to slightly lower performance loss than those of Table I for a fast-fading channel model. In fact, one demapping iteration with high-diversity channel model leads to more error correction compared to one iteration executed with limited-diversity one. Conducted simulations with block-fading channel model have confirmed this result.

Using this technique, latency and complexity issues caused by the TBICM-ID-SSD are reduced. Two feedbacks to the demapper with the associated delays, computations, and memory accesses are eliminated. It is worth noting that the proposed new scheduling does not have any impact on the receiver area (logic or memory). This scheduling is applied on a TBICM-ID-SSD receiver and proposes complexity reduction in “temporal dimension” (which impacts power consumption, throughput, and latency). Complexity reductions will be evaluated and discussed in the next section.

## V. COMPLEXITY EVALUATION AND NORMALIZATION

The main motivation behind the conducted convergence speed analysis and the proposed technique for reducing the number of iterations is to improve the receiver implementation quality. In order to appreciate the achieved improvements, an accurate evaluation of the complexity in terms of number and type of operations and memory access

	Parameter	Number of bits
SISO demapper	Received complex input $(x_{r,q}^I, x_{r,q}^Q)$	(10,10)
	Coeff. Fading & Variance $(h_q'^2)/(\sigma^2)$	8
	Constellation complex symbol $(s_{r,j}^I, s_{r,j}^Q)$	(12,12)
	Euclidean distance $A_q$	19
SISO decoder	Received 4 LLRs	$4 \times 5$
	Branch metric $\gamma_k$	10
	State metric $\alpha_k, \beta_k$	10
	Extrinsic information $z$	10

TABLE II: Typical quantization values.

Arithmetic operations ( $n_2 \geq n_1$ )	Normalized arithmetic operations
1 $Add(n_1, n_2)$	$0.5 \times (n_1 + n_2 - 1)Add(1, 1)$
1 $Sub(n_1, n_2)$	$0.5 \times (n_1 + n_2)Add(1, 1)$
1 $Mul(n_1, n_2)$	$[(n_1 - 1)(n_2 - 1) + 1 - 0.5 \times n_1]Add(1, 1)$

TABLE III: Complexity normalization in terms of  $Add(1, 1)$ .

SISO rotated demapper with <i>a priori</i> input	Computation units	Number and Type of operations per modulated symbol per turbo demapping iteration
	Euclidean distance	$2^M Add(18, 18) + 2^{M+1} Sub(8, 10) + 2^{M+1} Mul(8, 8) + 2^{M+1} Mul(8, 10) + 2load(10) + (1 + 2^M)load(8)$
	<i>a priori</i> adder	$(2^M - 2)\{E[\frac{M-1}{2}]Add(8, 8) + E[\frac{M-1}{4}]Add(9, 9) + E[\frac{M-1}{8}]Add(10, 10) + MSub(8, 11) + MSub(11, 19)\} + Mload(8) + 2^M load(M)$ For QPSK $M * 2^M - 2Sub(11, 19) + Mload(8) + 2^M load(M)$
	Minimum finder	$MSub(8, 8) + M.2^M Sub(8, 19) + Mstore(8)$
SISO double binary turbo decoder	Computation units	Number and Type of operations per coded symbol per turbo decoding iteration
	Branch metric	$4Add(5, 5) + 38Add(5, 10) + 4Sub(5, 5) + 8load(5) + 6load(10)$
	State metric	$64Add(10, 10) + 48Sub(9, 9) + 8store(10)$
	extrinsic information	$32Add(10, 10) + 32Sub(9, 9) + 9Sub(10, 10) + 3Mul(4, 10) + 8load(10) + 5store(10)$

TABLE IV: Complexity computation summary.

is required. Such complexity evaluation is fair and generalized as it is independent from the architecture mode (serial or parallel) and remains valid for both of them. In fact, all architecture alternatives should execute the same number of operations (serially or concurrently) to process a received frame. In this section, we consider the two main blocks of the TBICM-ID-SSD system configuration which are the SISO demapper and the SISO decoder. The proposed evaluation considers the low complexity algorithms presented in section II. A typical fixed-point representation of channel inputs and various metrics is considered. Table II summarizes the total number of required quantization bits for each parameter.

#### A. Complexity evaluation of SISO demapper

The complexity of SISO demapping depends on the modulation order (in the context of the above fixed parameters). We will now consider the equations of subsection II-B to compute: (1) the required number and type of arithmetic computations and (2) the required number of read memory access (*load*) and write memory access (*store*). The result of this evaluation is summarized in Table IV and explained below. We use the following notation  $operation(NbOfBitsOfOperand1, NbOfBitsOfOperand2)$  for arithmetic operations, and  $load(NbOfBits)/store(NbOfBits)$  for read/write memory operations. Thus,  $add(8, 10)$  indicates an addition operation of two operands; one quantized on 8 bits and the second on 10 bits. Similarly,  $load(8)$  indicates a read access memory of 8-bit word length.

##### 1) Euclidean distance computation

For each modulated symbol (input of the demapper):

- One  $load(8)$  to access the fading channel coefficient normalized by the channel variance  $\frac{h_q'^2}{\sigma^2}$
- Two  $load(10)$  to access the channel symbols  $x_{r,q}^I$  and  $x_{r,q}^Q$ .
- For each one of the  $2^M$  symbols of the constellation  $(s_{r,j}^I, s_{r,j}^Q)$ :
  - Two  $load(8)$  to access the constellation symbols  $s_{r,j}^I$  and  $s_{r,j}^Q$
  - Two  $Sub(8, 10)$  to compute  $(x_{r,q}^I - s_{r,j}^I)$  and  $(x_{r,q}^Q - s_{r,j}^Q)$
  - Two  $Mul(8, 10)$  to multiply with the channel coefficients  $\frac{h_q'}{\sigma}$  and  $\frac{h_{q-1}'}{\sigma}$
  - Two  $Mul(10, 10)$  to compute the square of the results above



- One  $Add(18, 18)$  to realize the sum of the two Euclidean distance terms

## 2) **A priori adder**

For each modulated symbol (input of the demapper):

- $M \text{ load}(8)$  to access the *a priori* informations  $L_{apr, Dem}(c_{p,q})$
- For each one of the  $2^M$  symbols of the constellation  $(s_{r,j}^I, s_{r,j}^Q)$ , except two symbols corresponding to all zeros and all ones:
  - One  $\text{load}(M)$  to access constellation symbol bits  $c_{p,q}$ .  $k = 0, 1, \dots, M - 1$
  - $E[\frac{M-1}{2}] \text{ Add}(8, 8)$  to realize the sum of two  $L_{apr, Dem}(c_{p,q})$
  - $E[\frac{M-1}{4}] \text{ Add}(9, 9)$  to realize the sum of four  $L_{apr, Dem}(c_{p,q})$
  - $E[\frac{M-1}{8}] \text{ Add}(10, 10)$  to realize the sum of eight  $L_{apr, Dem}(c_{p,q})$
  - $M \text{ Sub}(8, 11)$  to subtract the LLR of the specific  $p^{th}$  bit and thus obtain  $B_{p,q}$
  - $M \text{ Sub}(11, 19)$  to realize  $A_q - B_{p,q}$

$E[x]$  represents here the ordinary rounding of the positive number  $x$  to the nearest integer.

However, for the simple QPSK modulation the above operations can be simplified as only 2 LLRs exist for one modulated symbol. In fact, in equation (6) there is no need to execute an addition followed by a subtraction of the same LLR. Thus, the total number of required arithmetic operations in this case is  $4\text{Sub}(11, 19)$ .

## 3) **Minimum finder**

For each one of the  $M$  bits per modulated symbol:

- $2^M \text{ Sub}(19, 19)$  to realize the two min operations of equation (4)
- One  $\text{Sub}(8, 8)$  to subtract the above found 2 minimum values resulting in the demapper extrinsic information
- One  $\text{store}(8)$  to store the extrinsic information value

## B. Complexity evaluation of SISO decoder

The SISO decoder complexity is composed of 3 principal units: branch metric, state metric, and extrinsic information functions. As for the SISO demapper, the result of the complexity evaluation is summarized in Table IV and explained below. As stated before, the considered turbo code is an 8-state double binary one. At the turbo decoder side, each double binary symbol should be decoded to take a decision over the 4 possible values (00, 01, 10, 11).

### 1) **Branch metrics ( $\gamma$ )**

For each coded symbol (input of the decoder):

- 4  $\text{load}(5)$  to access systematic and parity LLRs
- 3  $\text{load}(10)$  to access demapper normalized extrinsic informations
- 2  $\text{Add}(5, 5)$  and 2  $\text{Sub}(5, 5)$  to compute systematic and parity branch metrics  $\gamma_{11}^{Sys}$ ,  $\gamma_{10}^{Sys}$ ,  $\gamma_{11}^{Parity}$  and  $\gamma_{10}^{Parity}$
- 19  $\text{Add}(5, 10)$  to compute branch metrics  $\gamma_k$  and  $\gamma_k^{Sys} + \gamma_k^{Parity}$

Operations above should be multiplied by 2 to generate forward and backward branch metrics.

### 2) **State metrics ( $\alpha, \beta$ )**

For each coded symbol (input of the decoder):

- 32  $\text{Add}(10, 10)$  to compute  $\alpha_{k-1}(s') + \gamma_k(s', s)$  for the 32 trellis transitions (8-state double binary trellis)
- 24  $\text{Sub}(9, 9)$  to realize the 8 max (4-input) operations of equation (7). In fact, 1 max (N-input) can be implemented as N-1 max (2-input) operations. 1 max (2-input) corresponds to 1  $\text{Sub}$

- 8 *store*(10) to store computed state metrics only for left butterfly algorithm

Operations above should be multiplied by 2 to generate forward  $\alpha$  and backward  $\beta$  state metrics.

### 3) **Extrinsic information** ( $z$ )

For each coded symbol (input of the decoder):

- 8 *load*(10) to access state metric values
- 32 *Add*(10, 10) to compute the second required addition operation in equation (10) for the 32 trellis transitions
- 28 *Sub*(9, 9) to realize the 4 max (8-input) operations of equation (10)
- 4 *Sub*(10, 10) to subtract symbol-level intrinsic information from the computed soft value (generating symbol-level extrinsic information)
- 8 *Sub*(9, 9) and 4 *Sub*(10, 10) to realize the 8 max (2-input) operations and compute 4 bit-level (systematic and parity) extrinsic information as demapper *a priori* information (equations (12) and (13)). This computation is done only for one of the two SISO decoders
- 4 *store*(10) to store the computed bit-level (systematic and parity) extrinsic information
- 3 *Sub*(10, 10) to normalize symbol-level extrinsic information by subtracting the one related to decision 00
- 3 *Mul*(4, 10) to multiply the symbol-level extrinsic information by a scaling factor SF
- 3 *store*(10) to store the computed DEC<sub>1</sub> symbol-level extrinsic information as DEC<sub>2</sub> *a priori* information

### C. Complexity normalization

The above conducted complexity analysis exhibits different arithmetic and memory operation types and operand sizes. In order to provide a fair evaluation of the improvement in complexity and memory access with the technique proposed in section IV, complexity normalization is necessary.

For arithmetic operations, normalization can be done in terms of 2-input one bit full adders (*Add*(1, 1)). Each one of the adders, subtractors, and multipliers can be converted into an equivalent number of *Add*(1, 1). For adders and subtractors, bit-to-bit half and full adders are used and generalized for operand sizes  $n_1$  and  $n_2$ . Obtained formulas are summarized in Table III with simple, yet accurate, analysis of all corner cases. Similarly, multiplication operations are normalized using successive addition operations. Memory access operation of  $m$  word of size  $n$  are normalized to one memory access operation of  $m \times n$  bits.

Applying the proposed complexity normalization approach to Table IV leads to the results summarized in Table V.

## VI. DISCUSSIONS AND ACHIEVED GAINS

This section evaluates and discusses the achieved complexity reductions using the proposed original iteration scheduling of TBICM-ID-SSD at different modulation orders and code rates. As concluded in section IV, two demapping iterations can be eliminated while keeping the number of turbo decoding iterations unaltered. Overall, this will lead to a reduction corresponding to two times the execution of the SISO demapping function. Besides the fact that the obtained results will depend on the modulation order and code rate, a third parameter should be considered regarding the iterative demapping implementation choice. In this regard, two configurations should be analyzed. In the first configuration, denoted CASE 1, the Euclidean distances are re-calculated at each demapping iteration. While in the second configuration, denoted CASE 2, the computation of the Euclidean distances are done only once, at the first iteration, then stored and reused in later demapping iterations. Thus, CASE 1 implies higher arithmetic computations, however less memory access, than CASE 2.

SISO rotated demapper with <i>a priori</i> input	Computation units	Number and Type of operations per modulated symbol per turbo demapping iteration
	Euclidean distance	$123.75 \cdot 2^{M+1} \text{Add}(1, 1) + \text{load}(28 + 2^{M+3})$
	<i>a priori</i> adder	$(2^M - 2) \{ 7.5E[\frac{M-1}{2}] + 8.5E[\frac{M-1}{4}] + 9.5E[\frac{M-1}{8}] + 24.5.M \} \cdot \text{Add}(1, 1) + \text{load}(8M) + \text{load}(M2^M)$ For QPSK $15.M(2^M - 2) \cdot \text{Add}(1, 1) + \text{load}(8.M + M.2^M)$
	Minimum finder	$(8 + 13.5 \cdot 2^M)M \text{Add}(1, 1) + \text{store}(8.M)$
SISO double binary turbo decoder	Computation units	Number and Type of operations per coded symbol per turbo decoding iteration
	Branch metric	$304 \text{Add}(1, 1) + \text{load}(100)$
	State metric	$1040 \text{Add}(1, 1) + \text{store}(80)$
	extrinsic information	$760 \text{Add}(1, 1) + \text{load}(80) + \text{store}(50)$

TABLE V: Complexity computation summary after normalization.

Modulation scheme	CASE1 (With recomputed Euclidean distances)						CASE2 (With stored Euclidean distances)					
	$R_c = 1/2$			$R_c = 6/7$			$R_c = 1/2$			$R_c = 6/7$		
	Complexity Reduction			Complexity Reduction			Complexity Reduction			Complexity Reduction		
	<i>arith</i>	<i>load</i>	<i>store</i>	<i>arith</i>	<i>load</i>	<i>store</i>	<i>arith</i>	<i>load</i>	<i>store</i>	<i>arith</i>	<i>load</i>	<i>store</i>
QPSK	11.9%	10.6%	3.7%	8.2%	7.1%	2.2%	2.5%	12%	3.4%	1.6%	8.2%	2.1%
QAM16	20.3%	13.7%	3.7%	15.9%	9.7%	2.2%	11.6%	18.1%	3.1%	8.3%	13.4%	2%
QAM64	27.9%	21.4%	3.7%	25%	17.1%	2.2%	21.8%	26.5%	2.5%	18.5%	22.3%	1.7%
QAM256	31.6%	28.4%	3.7%	30.5%	25.7%	2.2%	27.6%	32.2%	1.5%	26.2%	30%	1.2%

TABLE VI: Reduction in number of operations, read/write access memory comparing "4IDem\_2EIDec" to "6IDem" for different modulation schemes and code rates.

Using the normalized complexity evaluation of Table V, achieved gains comparing 4IDem\_2EIDec to 6IDem for all configurations are summarized in Table VI. In the following we will explain first how these values are computed and then discuss the obtained results.

In fact, considering the code rate  $R_c$  and the number of bits per symbol  $M$ , the relation between the number of double binary coded symbols ( $N_{CodedSymb}$ ) and the corresponding number of modulated symbols ( $N_{ModSymb}$ ) can be written as follows.

$$N_{ModSymb} = \frac{2 \cdot N_{CodedSymb}}{M \cdot R_c} \quad (15)$$

The complexity reduction ( $G$ ) corresponds to the ratio between the complexity of two SISO demapping executions and the complexity of the original TBICM-ID-SSD configuration. If the original TBICM-ID-SSD configuration requires  $N_{it}$  iterations to process a frame composed of  $N_{ModSymb}$  modulated symbols (equivalent to  $N_{CodedSymb}$  coded symbols), the complexity reduction can be approximated by the following expression.

$$G = \frac{2 \cdot F_{Dem}(M) \cdot N_{ModSymb}}{N_{it} \cdot F_{Dem}(M) \cdot N_{ModSymb} + N_{it} \cdot F_{Dec} \cdot N_{CodedSymb}} \quad (16)$$

where  $F_{Dem}$  designates the complexity of SISO demapper which depends on the constellation size and  $F_{Dec}$  designates the complexity of SISO decoder. When converting in this equation the number of modulated symbols into equivalent coded symbols using equation (15), we obtain the following equation.

$$G = \frac{2 \cdot F_{Dem}(M)}{N_{it} \cdot F_{Dem}(M) + N_{it} \cdot F_{Dec} \cdot \frac{M \cdot R_c}{2}} \quad (17)$$

This last equation has been used to obtain individually the complexity reductions in terms of arithmetic, read memory access, and write memory access operations of Table VI, for  $N_{it} = 6$ . For CASE 1, results show increased benefits in terms of number of arithmetic operations (up to 31.6%) and read memory accesses (up to 28.6%) with higher modulation orders. This can be easily predicted from equation (17) as the value of  $F_{Dem}$  increases with the constellation size. The equation shows also that the higher the code rate is, lower the benefits are.

On the other hand, the improvement in write memory access (3.7% for  $R_c = 1/2$  and 2.2% for  $R_c = 6/7$ ) is low and constant for all modulation orders. In fact, in Table VI the single memory *store* term which depends on the modulation order is  $store(8.M)$  for the minimum finder computation. This term is required per modulated symbol and when converted to the equivalent number per coded symbol (equation (15)) for a fixed code rate a constant value, independent from  $M$ , is obtained.

Similar behavior is shown for CASE 2, except for two points. The first one concerns the improvements in arithmetic operations and read memory accesses. In fact, compared to CASE 1, this configuration implies less arithmetic and more memory access operations which lead to less benefits for the former and more benefits for the latter (equation (17)). The second point concerns the improvement in write memory access. In fact, besides the term  $M \times 8bits$ , a value of  $19 \times 2^M$  is required only for the first iteration to store the  $2^M$  Euclidean distances quantized on 19 bits each. This added value is much higher in comparison to the reduced  $M \times 8bits$  write memory access. Therefore the improvement in write access memory operations will be less for higher constellation sizes (down to 1.2%).

It is worth noting that applying the proposed scheduling combined with an early stopping criteria might diminish the benefit from the scheduling, but at the cost of an additional complexity.

## VII. CONCLUSION

Convergence speed analysis is crucial in TBICM-ID-SSD systems in order to tune the number of iterations to be optimal when considering the practical implementation perspectives. Conducted analysis has demonstrated that omitting two turbo demodulation iterations without decreasing the total number of turbo decoding iterations leads to promising complexity reductions while keeping error rate performance almost unaltered. A maximum loss of 0.15 dB is shown for all modulation schemes and code rates in a fast-fading channel with and without erasure. The number of normalized arithmetic operations is reduced from 8.2% for QPSK configuration to  $\frac{2}{N_{it}}\%$  for QAM256 (e.g. for  $N_{it} = 6$  this gives a reduction of 33.3%). Similarly, the number of read access memory is reduced in a range between 8.2% to  $\frac{2}{N_{it}}\%$ . This complexity reduction improves significantly latency and power consumption, and thus paves the way towards the adoption of TBICM-ID-SSD hardware implementations in future wireless receivers. Future work targets the extension of this analysis to other baseband iterative applications and its integration into available hardware prototypes.

## REFERENCES

- [1] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," in *IEEE International Symposium on Information Theory*, 1997.
- [2] X. Li and J. Ritcey, "Bit-interleaved coded modulation with iterative decoding," *IEEE Communications Letters*, vol. 1, no. 6, pp. 169–171, 1997.
- [3] A. Chindapol and J. Ritcey, "Design, analysis, and performance evaluation for BICM-ID with square QAM constellations in rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 5, pp. 944–957, 2001.
- [4] I. Abramovici and S. Shamai, "On turbo encoded BICM," in *Annales des telecommunications*, vol. 54, 1999, pp. 225–234.
- [5] C. Abdel Nour and C. Douillard, "On lowering the error floor of high order turbo BICM schemes over fading channels," in *IEEE Global Telecommunications Conference, GLOBECOM'06*, 2006, pp. 1–5.
- [6] N. Kiyani and J. Weber, "Iterative demodulation and decoding for rotated MPSK constellations with convolutional coding and signal space diversity," in *IEEE Vehicular Technology Conference, VTC'07-Fall*, 2007, pp. 1712–1716.
- [7] J. Boutros and E. Viterbo, "Signal space diversity: a power- and bandwidth-efficient diversity technique for the rayleigh fading channel," *IEEE Transactions on Information Theory*, vol. 44, no. 4, pp. 1453–1467, 1998.
- [8] C. Abdel Nour and C. Douillard, "Improving BICM performance of QAM constellations for broadcasting applications," in *International Symposium on Turbo Codes and Related Topics*, 2008, pp. 55–60.
- [9] O. Muller, A. Baghdadi, and M. Jezequel, "Exploring Parallel Processing Levels for Convolutional Turbo Decoding," in *International Conference on Information and Communication Technologies, ICTTA'06*, vol. 2, 2006, pp. 2353–2358.
- [10] S. ten Brink, "Convergence of iterative decoding," *Electronics Letters*, vol. 35, no. 13, pp. 1117–1119, 1999.